School of Computing, Engineering and Built Environment

Big Data Landscape

Module Code: MMI226831

Coursework 1

Student Name: Arjun Paudel

Student Number: S2271954

Campus: London

"Except where explicitly stated all work in this document is my own."

Signed: Arjun **Date:** 24th November 2023

Contents

1 Task 1 1
1.1 Question Number 1 1
1.2 Question Number 2 1
1.3 Question Number 3 18
2 Task 2 18
2.1 Question Number 1 18
2.2 Question Number 2 20
3 Task 3 25

1 Task 1

1.1 Question Number 1

Details overview of google big query public dataset "**bigquery-public-data.fcc_political_ads**"

'**bigquery-public-data.fcc_political_ads**' is a Federal Communications Commission dataset that is aviable on google cloud big query which provide the details information regarding the political advertisement that has been advertise on television and radio across United States.

This dataset consists of 4 table and these tables are

1. **broadcast_tv_radio_station**:-
   This table contains the information about the radio station and television that had submitted different files, like city,state, party name, callsign, Nielsen dma, rfchannel and so on. In this table there are three field which value are required and all the other fields can be nullable and these required field are stationId, callSign and service. The total number of rows on this table is 17,910.

2. **content_info**:-
   This table provides all the information regarding the content that has been used during political advertisement. This table holds the information like who is the advertise of the content, from which candidate advertisement was done, first day and last day of the ad that aired in the broadcaster, file link to view the original files and so. Here only one field is required that is contentInfold which is a primary key to this table and all the other field can be nullable. The total number of rows on this table is 6,524.

3. **file_history**:-
   When the broadcaster uploaded the new version of advertisement, then the receipt of the previous version is store on this table. In this table only one field is required that is fileHistoryId which also the primary key and rest of all the field can be nullable. The total number of rows on this table is 1,553,461.

4. **file_record**:-
   This table hold the information regarding the originals files retrieved from the a Federal Communications Commission, with extra field added like as path to a plain text version of files. This table contains a one required field that is fileRecordId which is also a primary key for this table and rest of the field in this table can be nullable. The total number of rows in this table is 1,791,357.

To join the different table there are some filed which are common to each other. To join the broadcast_tv_radio_station, file_history, and file_record we can use stationId field. And also to join the file_record and content_info table we can rawFilePath field.

1.2 Question Number 2

Before finding any values or information from the dataset we have to check whether that dataset has quality of data. As we mentioned on topic 1.1, there are some field on this dataset which are required. So, first Checking if there any null value in required field all over the four tables.

1) First on the **broadcast_tv_radio_station** table was checked.

Code:

```
SELECT
  COUNT(*) AS TotalRows,
  COUNTIF(stationId IS NULL) AS MissingStationId,
  COUNTIF(callSign IS NULL) AS MissingCallSign,
  COUNTIF(service IS NULL) AS MissingService
FROM
  `bigquery-public-data.fcc_political_ads.broadcast_tv_radio_station`
```

This code is used to querying a dataset in Google Cloud BigQuery, where fcc_political_ads is a google bigquery public dataset and broadcast_tv_radio_station is a one table inside that dataset. '**Count(*)**' function is used to count the total number of rows in table and with the '**AS**' function is used to label the result as '**TotalRows**'. Like as '**COUNTIF(StationId is NULL) As MissingStationId'** is another query where countif function counts the number of rows where **stationId** column is null and labels the results as MissingStationId. And '**From**' function select the data from that particular dataset table. Always while running a query we have to put **project.dataset.table**.

Output:

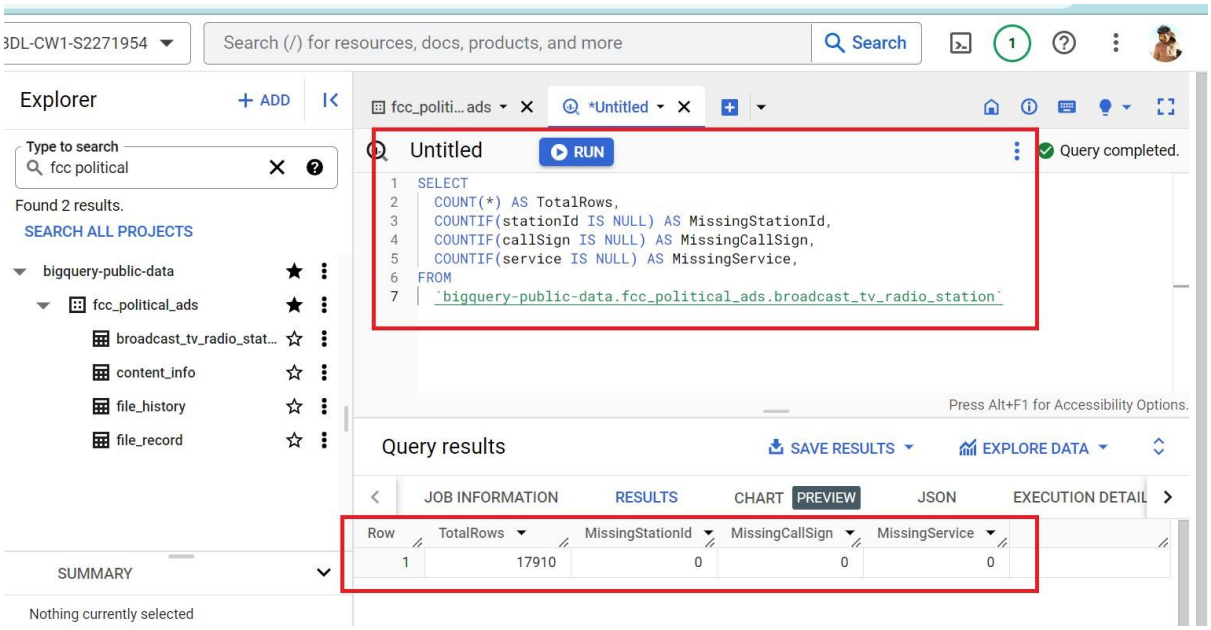

Figure 1:Result of query to find out the null values in broadcast_tv_radio_station table

2) On **content_info** table,

Code:

```
SELECT
  COUNT(*) AS TotalRows,
  COUNTIF(contentInfoId IS NULL) AS MissingContentInfoId,
FROM
```

```
  `bigquery-public-data.fcc_political_ads.content_info`
```

This code is used to querying a dataset from content_info table. '**Count(*)**' function is used to count the total number of rows in table and with the '**AS**' function is used to label the result as '**TotalRows**'. Like as '**COUNTIF(contentInfoId is NULL) As MissingContentInfoId'** is another query where countif function counts the number of rows where **contentInfoID** column is null and labels the results as MissingContentInfoId.

Output:

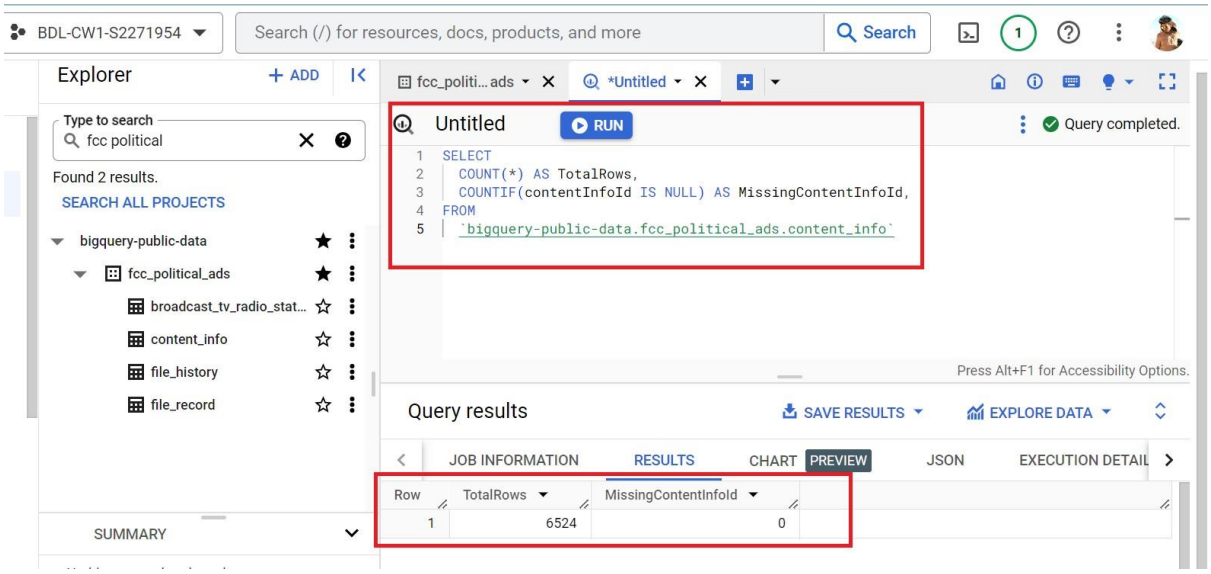

Figure 2:Result of query to find out the null values in content_info table

3) On **file_history** table

Code:

```
SELECT
  COUNT(*) AS TotalRows,
  COUNTIF(fileHistoryId IS NULL) AS MissingFileHistoryID,
FROM
  `bigquery-public-data.fcc_political_ads.file_history`
```

This code is used to querying a dataset from file_history table. '**Count(*)**' function is used to count the total number of rows in table and with the '**AS**' function is used to label the result as '**TotalRows**'. Like as '**COUNTIF(fileHistoryId is NULL) As MissingFileHistoryIDId'** is another query where countif function counts the number of rows where **fileHistoryId** column is null and labels the results as MissingFileHistoryID.

Output:

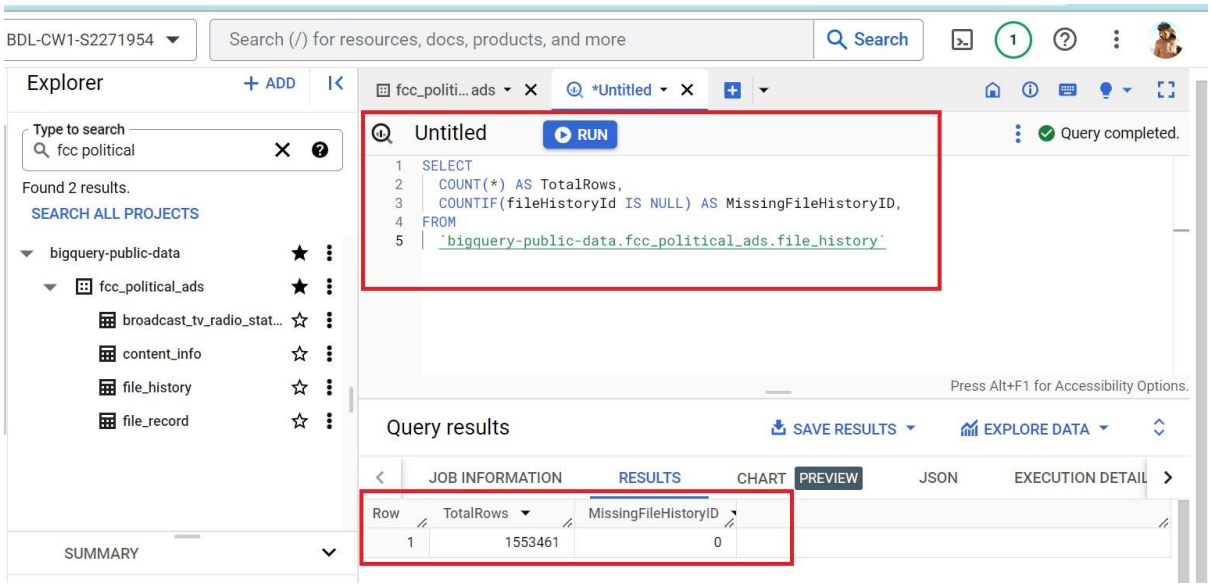

Figure 3:Result of query to find out the null values in file_history table

4) On **File_record** table

Code:

```
SELECT
  COUNT(*) AS TotalRows,
  COUNTIF(fileRecordId IS NULL) AS MissingFileRecordID,
FROM
  `bigquery-public-data.fcc_political_ads.file_record`
```

This code is used to querying a dataset from file_record table. '**Count(*)**' function is used to count the total number of rows in table and with the '**AS**' function is used to label the result as '**TotalRows**'. Like as '**COUNTIF(fileRecordId is NULL) As MissingFileRecordID'** is another query where countif function counts the number of rows where **fileRecordID** column is null and labels the results as MissingFileRecordID.

Output:

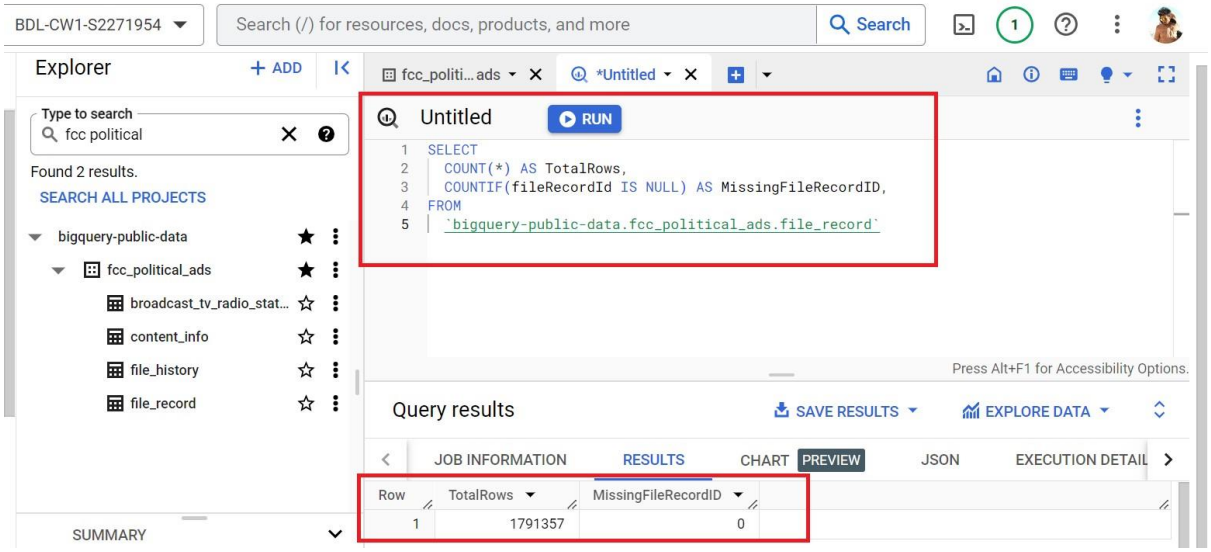

Figure 4:Result of query to find out the null values in file_record table

After finishing this validity some SQL queries were run to explore our dataset more.

Returns all the columns from broadcast_tv_radio_station.

Code:

```
SELECT
  *
FROM
  `bigquery-public-data.fcc_political_ads.broadcast_tv_radio_station`
```

This code retrives all the columns from broadcast_tv_radio_station table in the dataset of bigquery-public-data.fcc_political_ads.

Output:

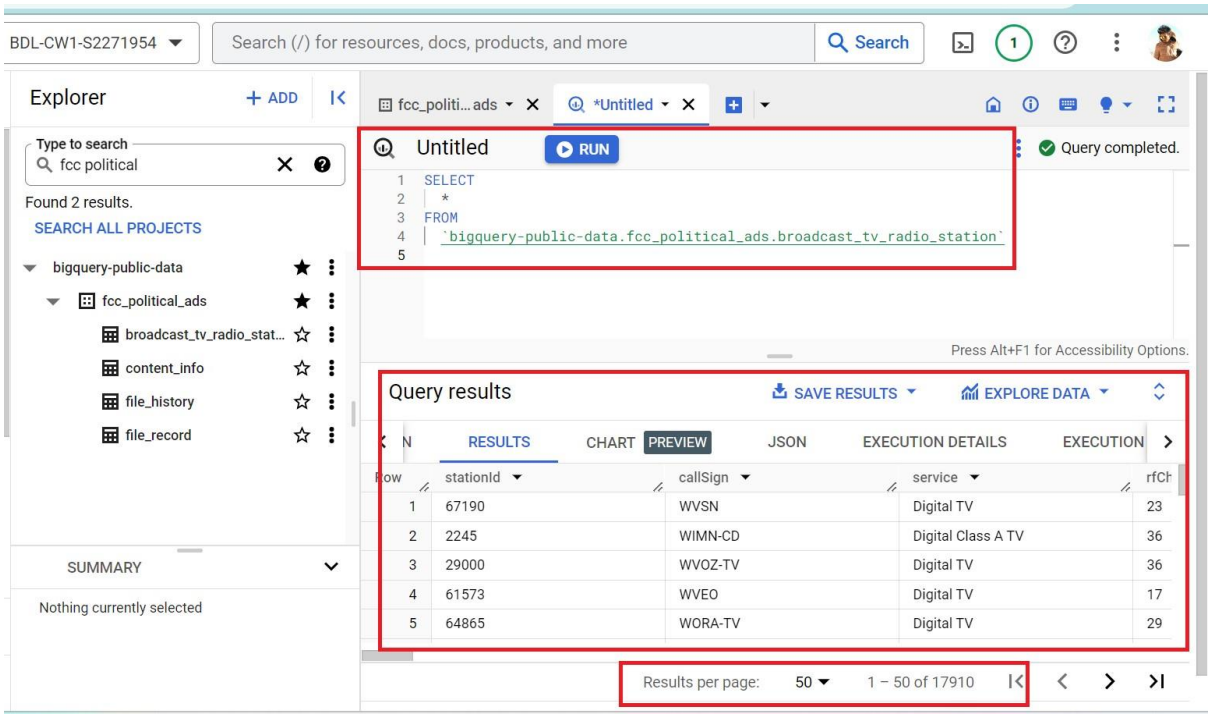

Figure 5:Displaying all the details form one table

Display only one column as a result.

Code:

```
SELECT
  stationId
FROM
  `bigquery-public-data.fcc_political_ads.broadcast_tv_radio_station`
```

This code only selects a single column from the entire table and display that result.

Output:

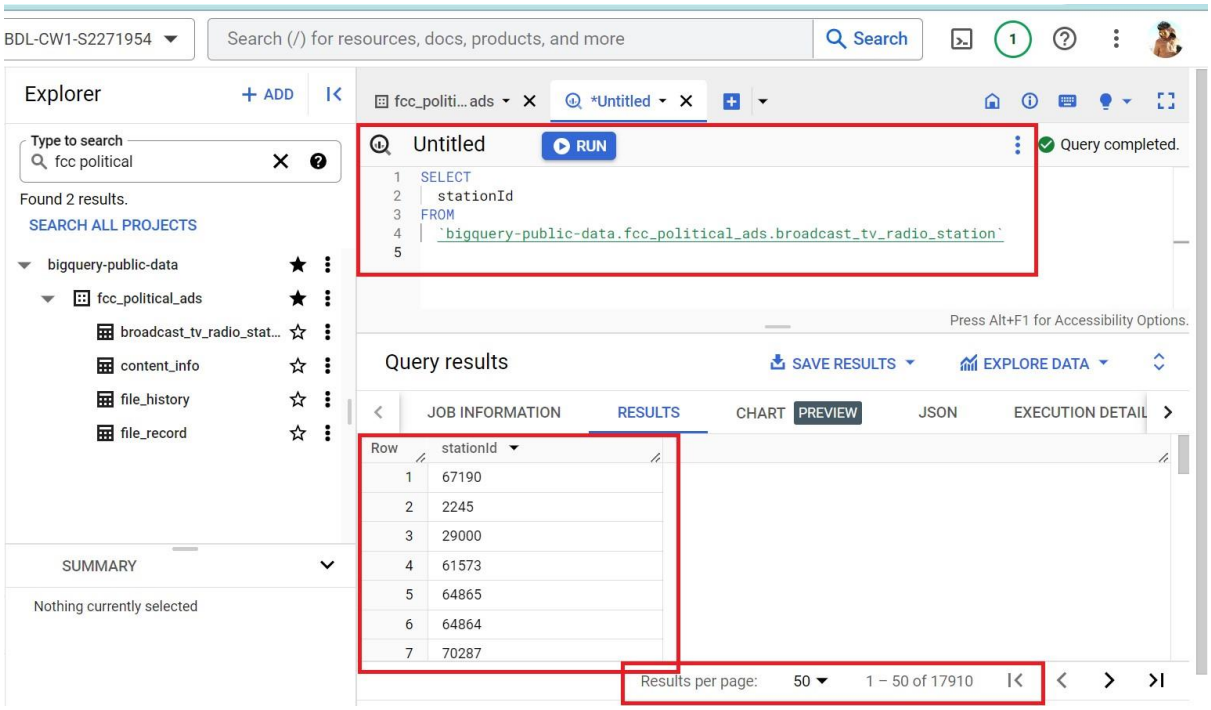

Figure 6: Selecting single column only

The code display result of all the 17910 rows while executing. But if we need only few row as a result then we can do this.

Code:

```
SELECT
  stationId
FROM
  `bigquery-public-data.fcc_political_ads.broadcast_tv_radio_station`
LIMIT
  7
```

This code only selects a single column from the entire table and display that result of 7 because Limit function allows only display those number that was given to it.

Output:

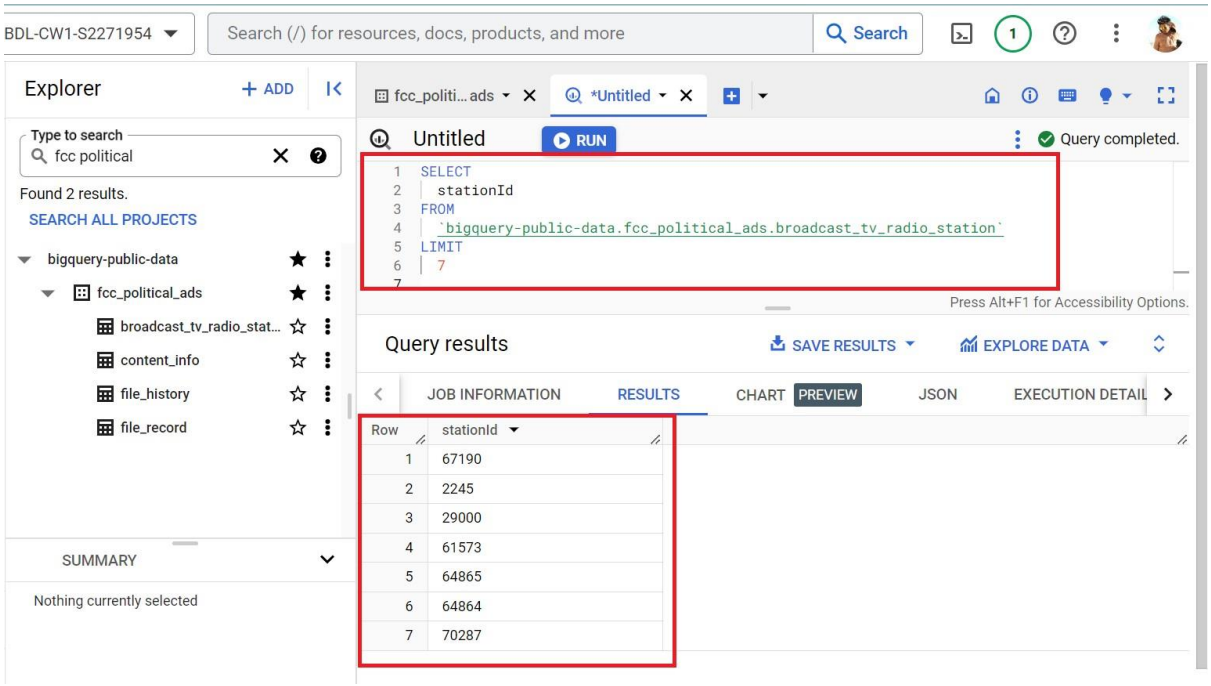

Figure 7: Selecting single column with limit

When we obtain the result, the result always come in unsorted way, but we can put a sql function to sort it.

Code:

```
SELECT
  stationId
FROM
  `bigquery-public-data.fcc_political_ads.broadcast_tv_radio_station`
ORDER BY
  stationId DESC
LIMIT
  7
```

Here Order By function helps to sort the stationId data into descending order

Output:

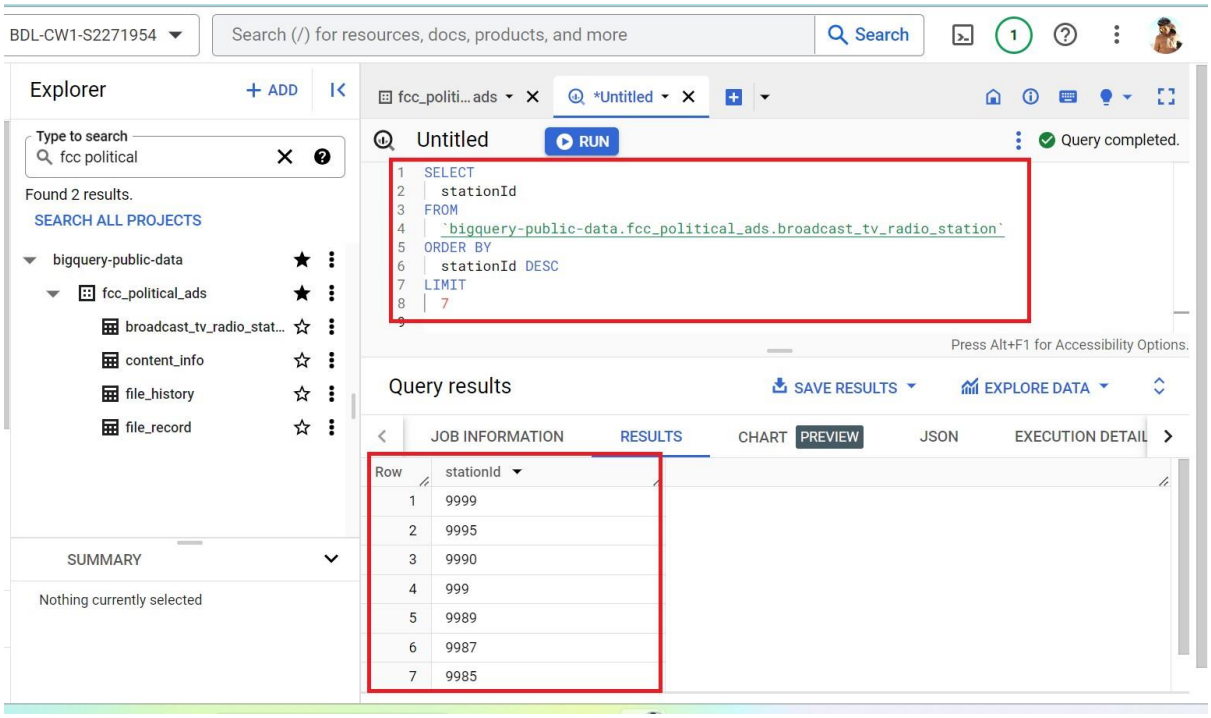

Figure 8:Using order by function

When we obtain the result in sorted order we still put some sql function to formatting data.

Code:

```
SELECT
  FORMAT("%'d", CAST(stationId AS INT64))
FROM
  `bigquery-public-data.fcc_political_ads.broadcast_tv_radio_station`
ORDER BY
  stationId DESC
LIMIT
  7
```

Here **FORMAT** function is used to format the result with commas for thousands. And this **CAST** function helps to change the string into integer. As in our table this stationID filed is define as string so we use the cast function to change it integer before formatting.

Output:

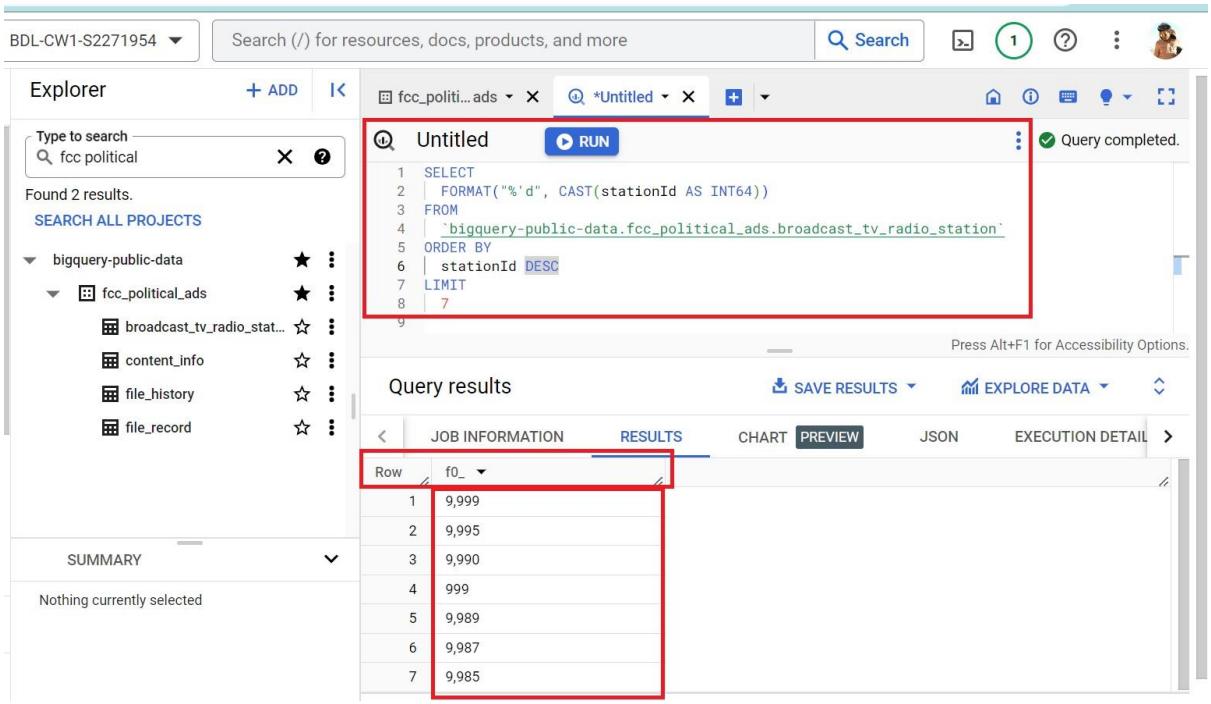

Figure 9:Using format function to formatting text

Wait in output of figure 9 there is result where column name has been changed into f0_ so to change the column name we introduce a new function '**AS**'.

Code:

```
SELECT
  FORMAT("%'d", CAST(stationId AS INT64)) AS station
FROM
  `bigquery-public-data.fcc_political_ads.broadcast_tv_radio_station`
ORDER BY
  stationId DESC
LIMIT
  7
```

This AS function rename the result of the query for stationID column as station.

Output:

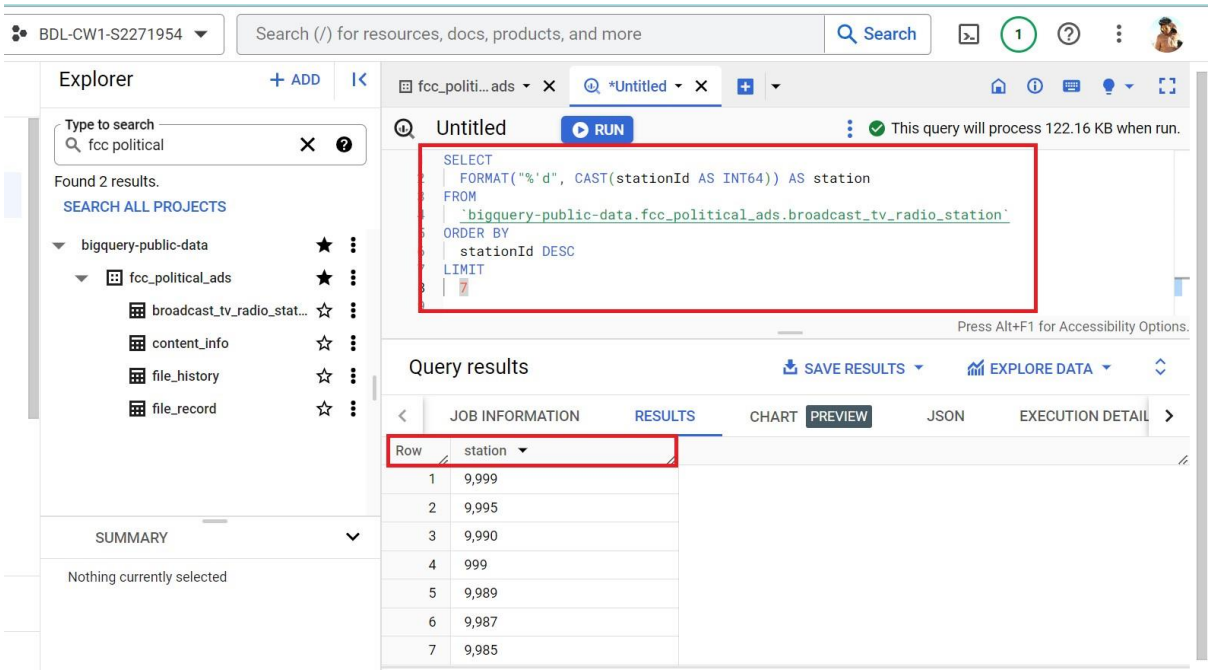

Figure 10: Assigning new name with the help of AS function

Now Let's play with our actual data. Here Find the data from broadcast_tv_radio_station where the partyzip2 is 5000?

Code:

```
SELECT
  partyZip2
FROM
  `bigquery-public-data.fcc_political_ads.broadcast_tv_radio_station`
WHERE
 partyZip2 = '5000'
LIMIT
  7
```

Adding where clause to filter the rows returned by a query based on specified condition. In this code it only filter the first 7 data from the broadcast_tv_radio_station table where partyzip2 code is 5000.

Output:

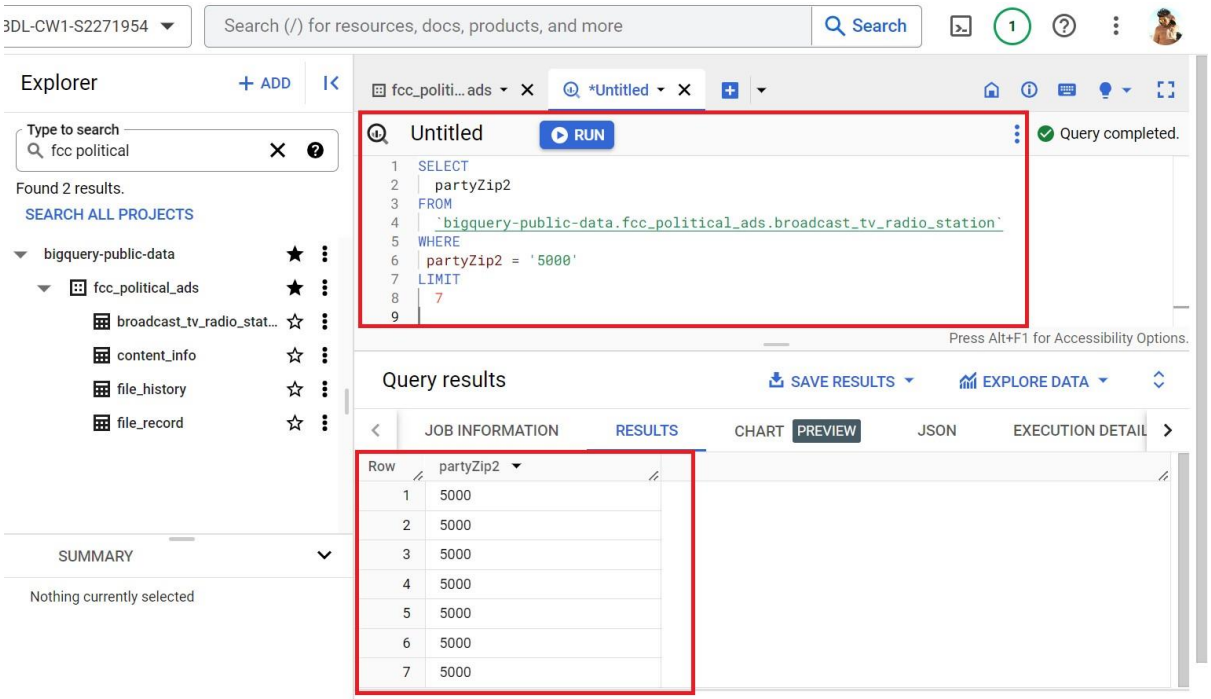

Figure 11: Finding the data from broadcast_tv_radio_station where the partyzip2 is 5000

Another question to answer- find the data where partyzip2 is 5000 along with the stationId?

Code:

```
SELECT
  partyZip2 AS ZIP,
  stationId AS Station
FROM
  `bigquery-public-data.fcc_political_ads.broadcast_tv_radio_station`
WHERE
  partyZip2 = '5000'
LIMIT
  7
```

This query gives the result when that condition is met along with the stationId .

Output:

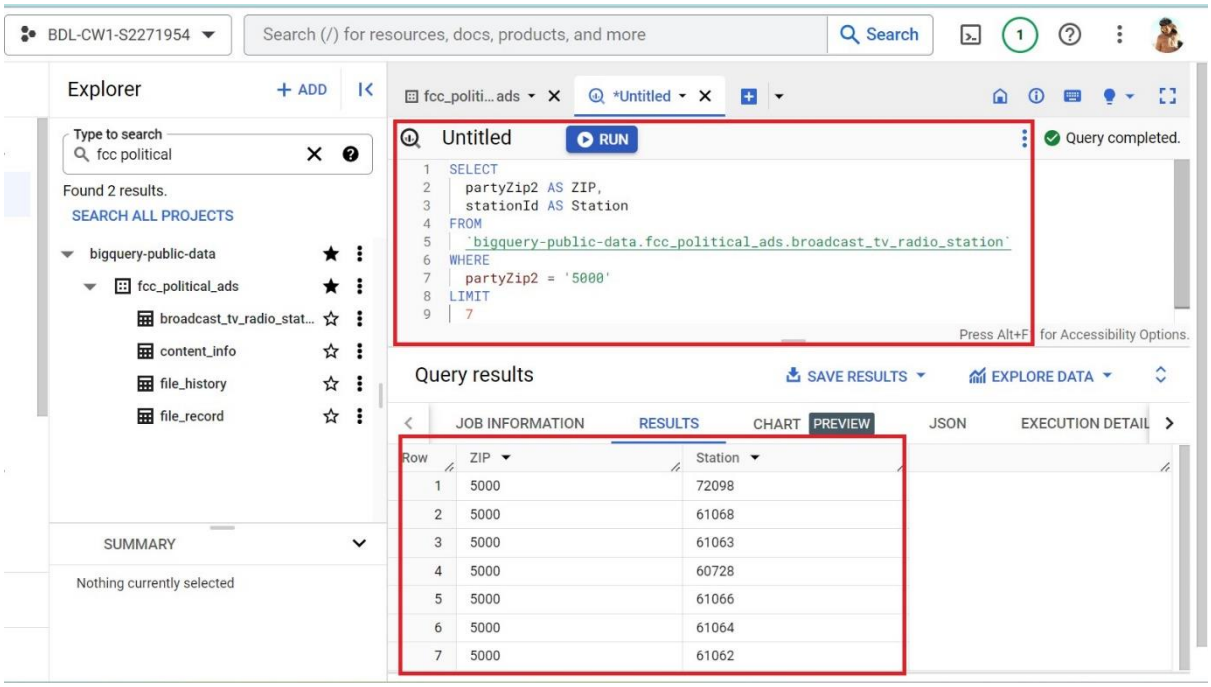

Figure 12: Finding the data where partyzip2 is 5000 along with the stationId

Find the total number of advertiser and total, average, maximum, minimum spend in ads through out the whole dataset.

Code:

```
SELECT
  COUNT(advertiser ) AS Total_Advertiser,
  SUM(grossSpend) AS Total_Spend,
  AVG(grossSpend) AS Average_grossSpend,
  MAX(grossSpend) AS Maximum_grossSpend,
  MIN(grossSpend) AS Minimum_grossSpend
FROM
  `bigquery-public-data.fcc_political_ads.content_info`
```

This code will run a query where COUNT function gives total number of advertiser, SUM function gives the total money spend in ads, AVG function gives the average money spend on the ads, MAX function is used to give the maximum amount spend on a single ads and min function gives a result on the minimum spends on the ads. Result of money is measured on cent.

Output:

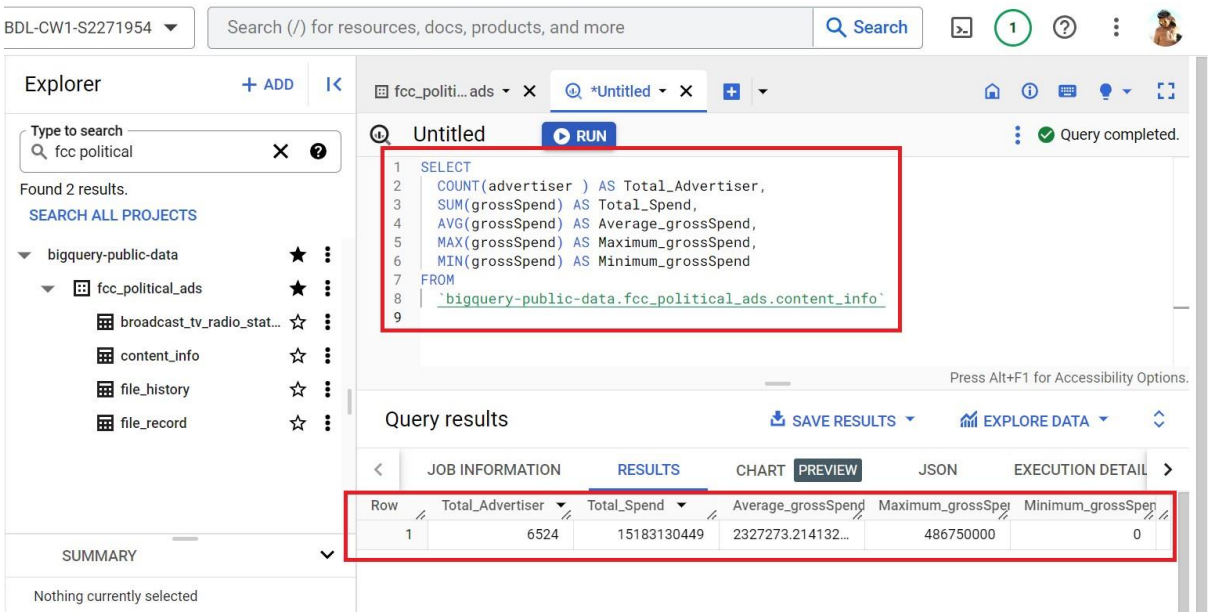

Figure 13: Finding the total number of advertiser and total, average, maximum, minimum spend in ads

Find the Total occurrence of each advertiser in dataset?

Code:

```
SELECT
  advertiser AS Total_Advertiser,
  COUNT(advertiser) AS advertiser_count
FROM
  `bigquery-public-data.fcc_political_ads.content_info`
GROUP BY
  advertiser
ORDER BY
  advertiser_count DESC
```

Here first query selects the advertiser name field and save the result as total_adversiter and count function is used to count the occurrence of that particular advertiser and save a result as advertiser_count, whereas group by functions is used to group the result according to the advertiser fields and order by function is used to display the result in descending order account to the advertiser_count.

Output:

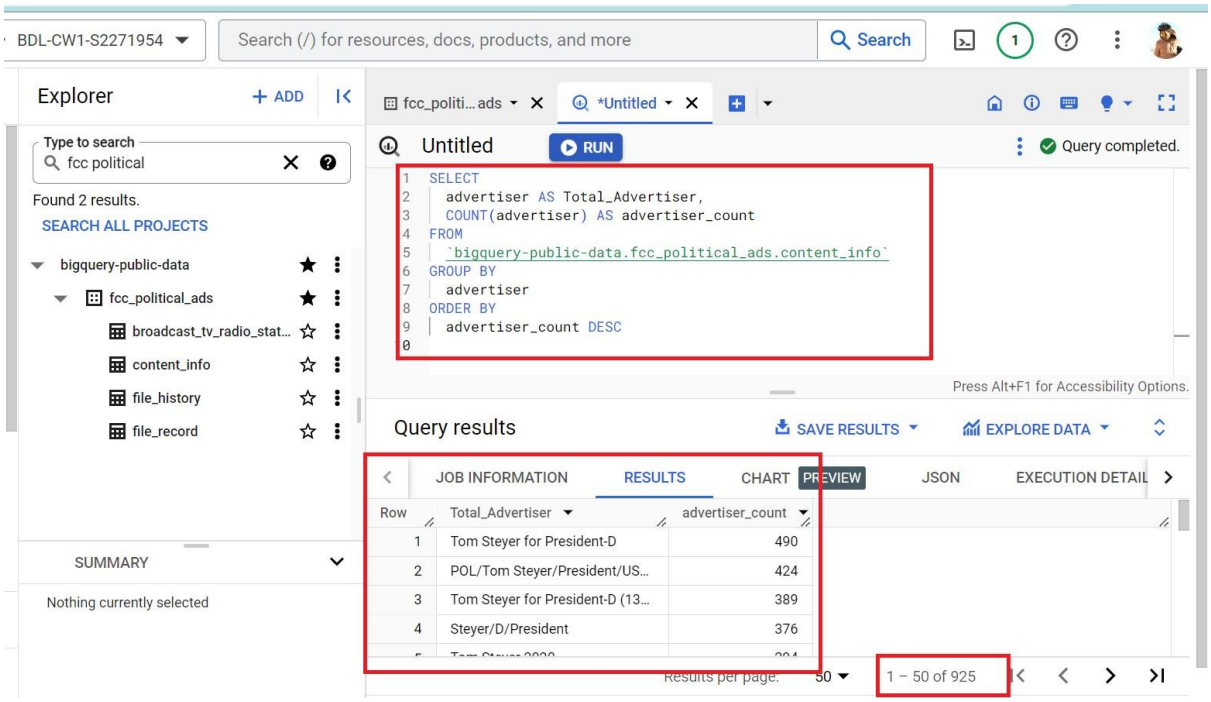

Figure 14: Finding the Total occurrence of each advertiser in dataset

Find the data for the specific advertiser and order it by grossSpend?

Code:

```
SELECT
  *
FROM
  `bigquery-public-data.fcc_political_ads.content_info`
WHERE
  advertiser = 'Buttigeig / D / President'
ORDER BY
  grossSpend ASC
```

This code helps to execute all the details of advertiser name ‘Buttigeig / D / President’ and order the result by grossSpend in ascending order.

Output:

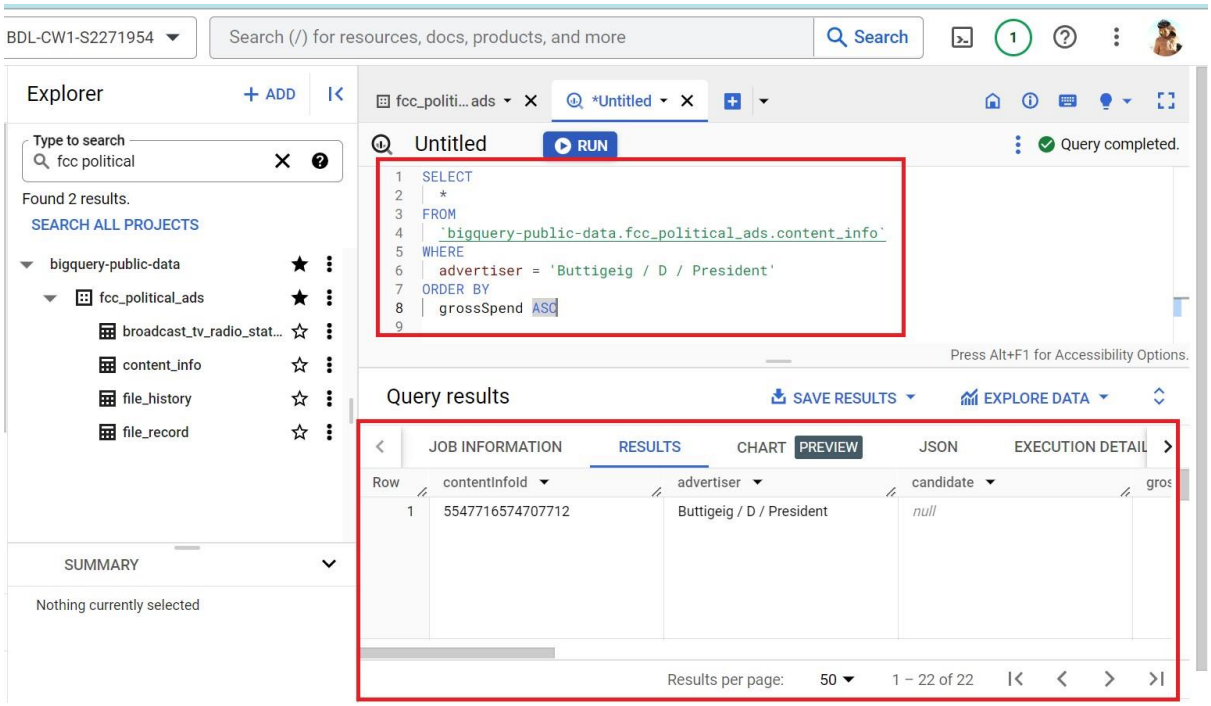

Figure 15:Finding the data for the specific advertiser and order it by grossSpend

Find all the files which has include 2019 in their name?

Code:

```
SELECT
  fileName
FROM
  `bigquery-public-data.fcc_political_ads.file_record`
WHERE
LOWER(fileName) LIKE '%2019'
limit 100
```

This code executes all the filename where the conditions is like to 2019.

Output:

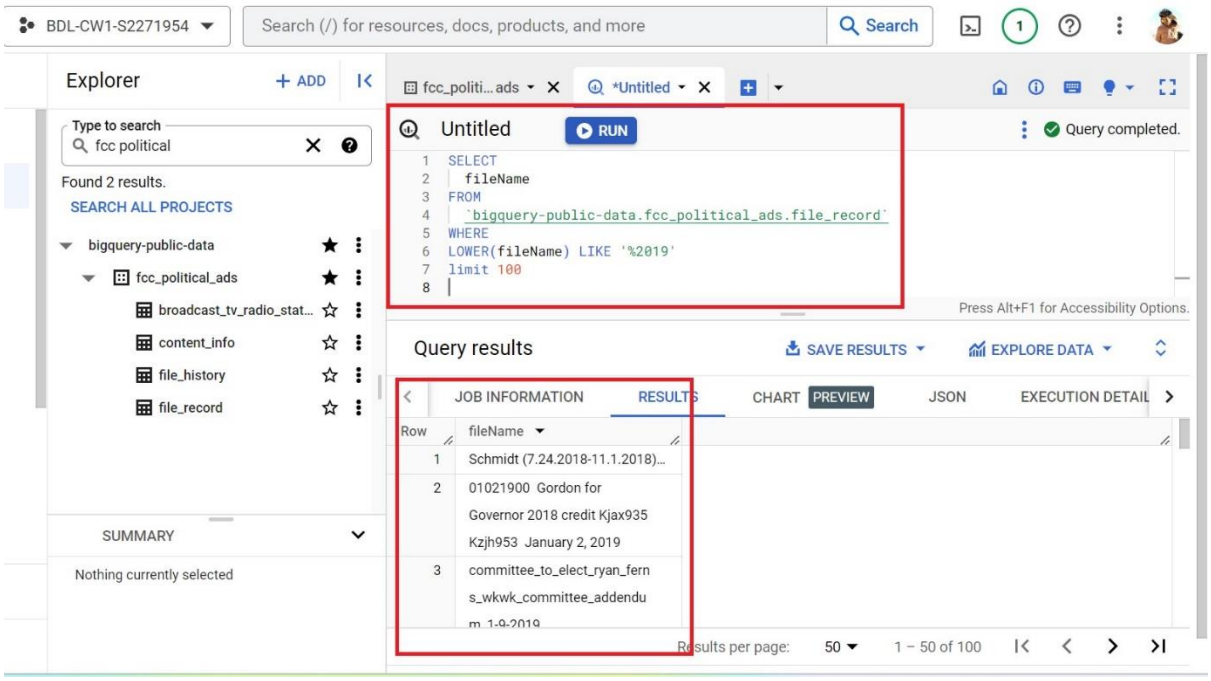

Figure 16:Finding all the files which has include 2019 in their name

Find which advertiser spends the most amount of money in ads?

Code:

```
SELECT
  advertiser,
  SUM(grossSpend) AS Totalspend
FROM
  `bigquery-public-data.fcc_political_ads.content_info`
GROUP BY
  advertiser
ORDER BY
  Totalspend DESC
LIMIT
  1
```

This code executes the highest total amount of money spends by single advertiser throughout the dataset. Here sum function is used to get the grossSpend of each particular advertiser and display its result in descending order according to that Totalspend result.

Output:

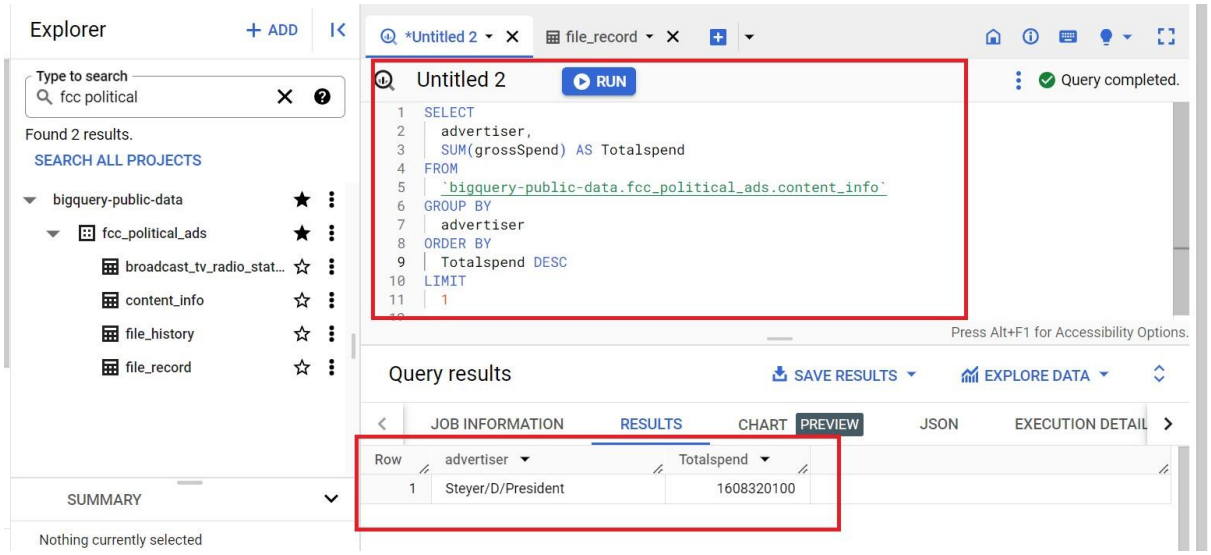

Figure 17:Finding which advertiser spends the most amount of money in ads

1.3 Question Number 3

From the query that are run in chapter 1.2 we can get the valuable information about the gross spend, highest spending advertiser, files name that has 2019 in their record and so on. From this query we can get those valuable information to run the election campaign through advertisement in radio and TV station to get the actual result or win the elections. Few more question has been answer from this query but the answer for that question that I wish to get hasn't been solve yet. Question that I wish to get from this query is which ten advertiser upload the most number of files in 2020. Answer of this question is further disused in another chapter below that is in task 2 question number 1.

2 Task 2

2.1 Question Number 1

Question 1: Which ten advertiser upload the most number of files in 2020?

Code:-

```
SELECT
  count (record.FileID) AS file,
  b.advertiser
FROM
  `bigquery-public-data.fcc_political_ads.file_record` AS record
JOIN
  `bigquery-public-data.fcc_political_ads.content_info` AS b
USING
  (rawFilePath)
WHERE
  year = 2020
GROUP BY
  b.advertiser
ORDER BY
  COUNT(fileId) DESC
LIMIT
  10
```

This code is used to get the query from the given two table by using the function Join. But for joining these table there should be a another fields which are common in both. As we already discussed in chapter 1 about the overview of dataset, there we mentioned that there is field called as rawFilePath which is used to connect the two table that is files_record and content_info. Here in this code first count function count the total number of files from the file_record table which is label as record and save that count result as file, and b.advertiser select the advertiser name from the conten_info table. Using function is used to connect the both table with that unique field which is rawFilePath, there is a condition set where year is 2020 give the query result which is group with advertiser and order the result according to the fileId in descending order.

Output:

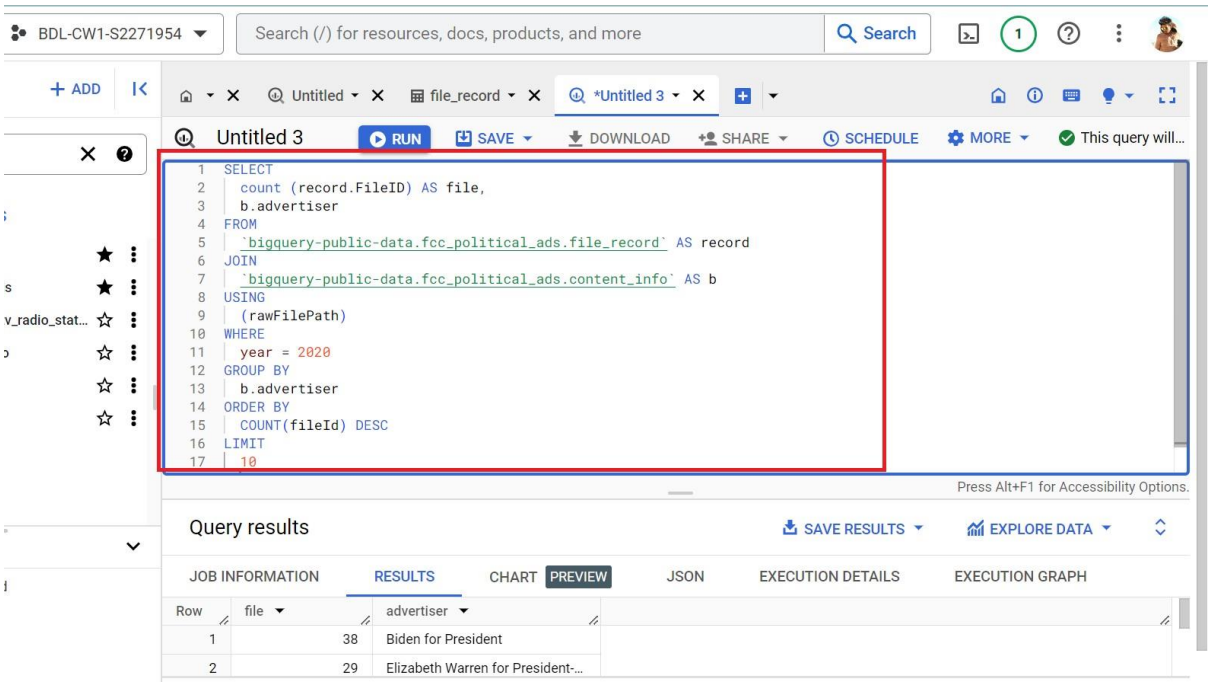

Figure 18:Which ten advertiser upload the most number of files in 2020

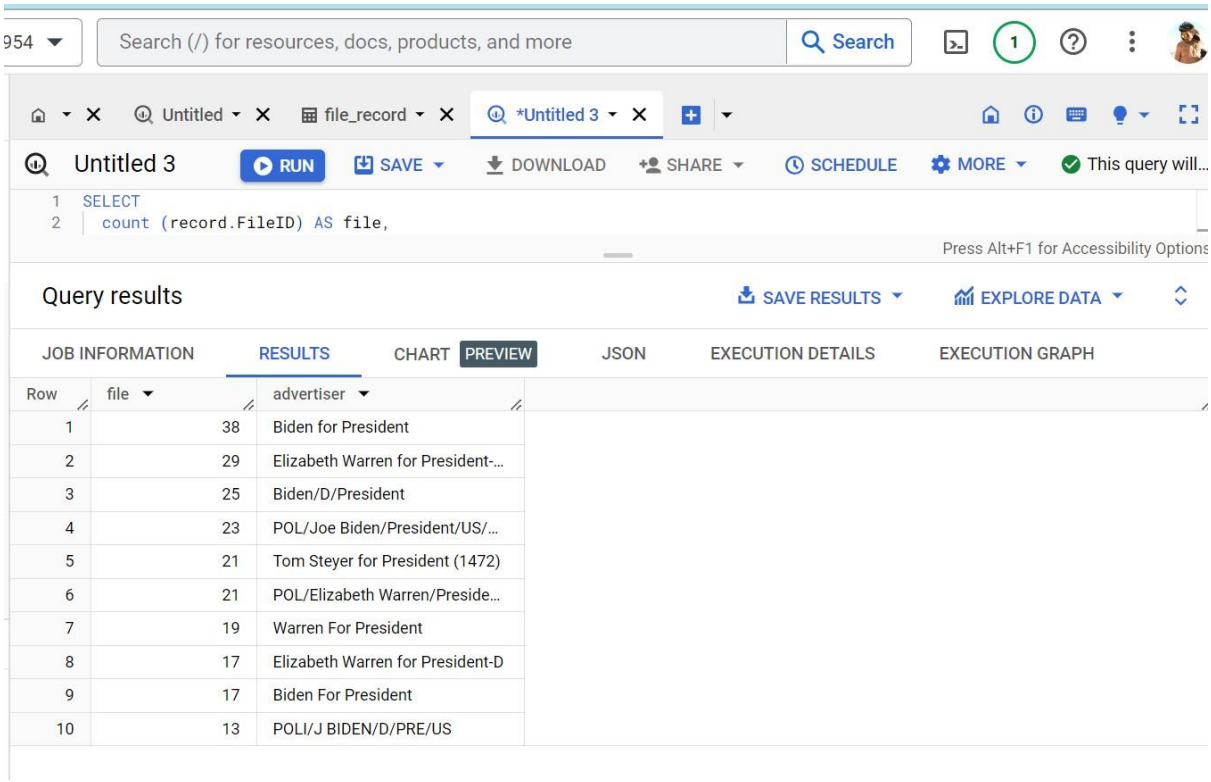

Figure 19: Result of ten advertiser that spend most amount of money in ads in 2020

2.2 Question Number 2

Now in the chapter we execute the same SQL code but in Google Colab by using both SQL and python. And also we create a google cloud bucket and store the result of query directly into that bucket.

First we have to authenticate to allow google colab to have an access to google cloud and google driver. This code will authenticate the author to use google cloud and google driver from google colab.

Code;

```
from google.colab import auth
auth.authenticate_user()
print('Authenticated')
```

Output:

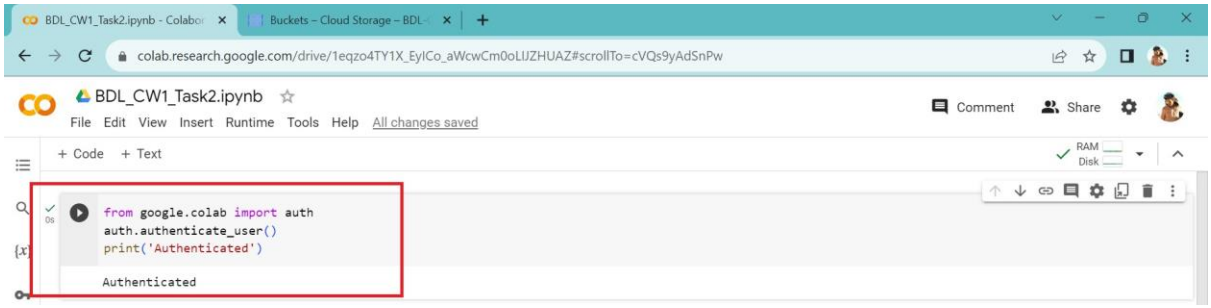

Figure 20: Authentication in Google colab

After allowing the authorization next step is to setting up the google cloud project ID. Google cloud project ID is unique ID for each project that are run in google cloud. This following code helps to setup our google cloud project ID.

Code:

```
project_id = 'bdl-cw1-s2271954'
!gcloud config set project {project_id}
```

Output:

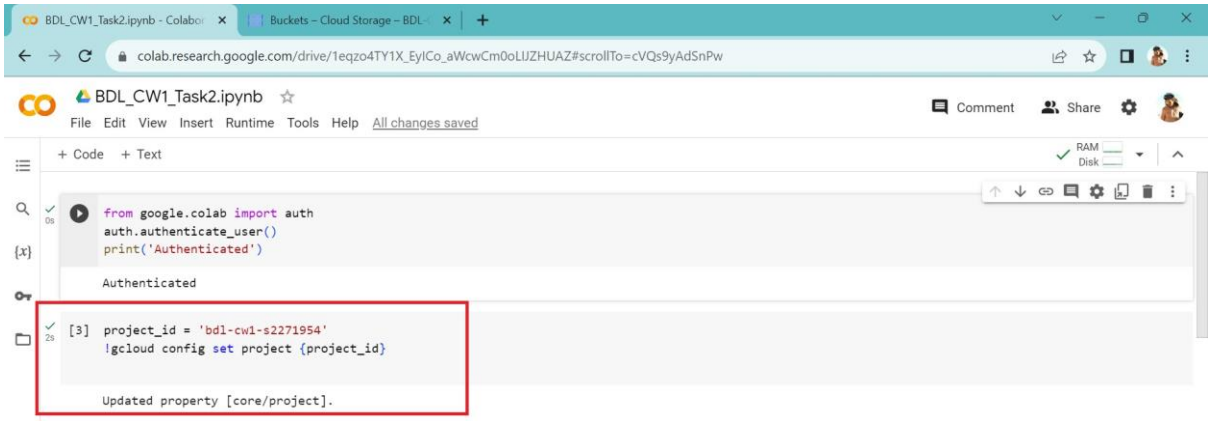

Figure 21: Connect colab notebook with google cloud project ID

Code:

After setting up a project id we have to create a unique bucket name. For this we have to import storage function from google cloud and define a storage client with our project id. Then naming a unique bucket name and store that information in cloud_storage_bucket_name. After that new line of code is execute to create the new bucket and also it gives the result where the bucket has created or not.

```
# Set up Google Cloud Storage client
from google.cloud import storage
storage_client = storage.Client(project=project_id)

# Create a unique name for Google Cloud Storage bucket
cloud_storage_bucket_name = 'S2271954_bucket'

# Create a new bucket
```

```
bucket = storage_client.create_bucket(cloud_storage_bucket_name)
print(f"Bucket {bucket.name} created.")
```

Output:

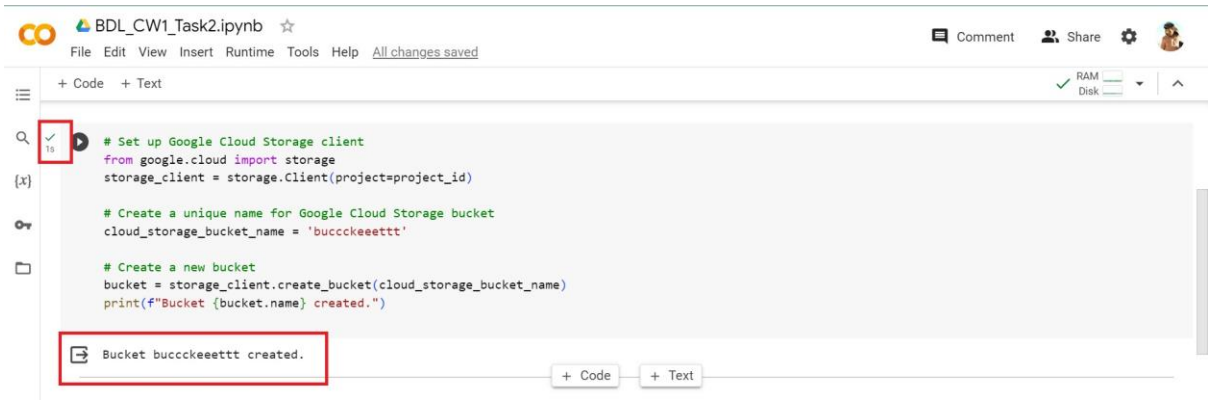

Figure 22: Creating bucket from google colab

Let see it into the google cloud console whether our bucket is created or not?

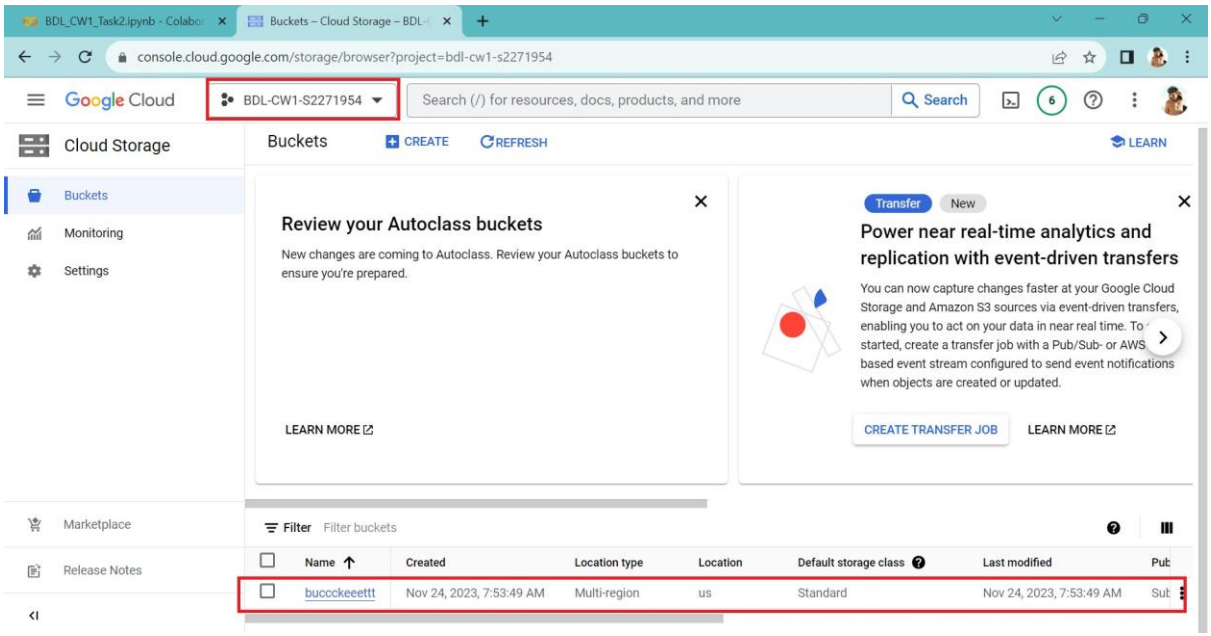

Figure 23: Bucket in google cloud

Code:

After creating bucket name its time run the sql queries. First bigquery library was import google cloud and also a pandas library for python code. First sql is run and save that result into dataframe which is result_df to save it as file in google bucket.

```
# Execute SQL code
from google.cloud import bigquery
import pandas as pd
```

```
# Set up BigQuery client
client = bigquery.Client(project=project_id)

# Write and execute SQL code
sql_query = """
SELECT
  count (record.FileID) AS file,
  b.advertiser
FROM
  `bigquery-public-data.fcc_political_ads.file_record` AS record
JOIN
  `bigquery-public-data.fcc_political_ads.content_info` AS b
USING
  (rawFilePath)
WHERE
  year = 2020
GROUP BY
  b.advertiser
ORDER BY
  COUNT(fileId) DESC
LIMIT
  10

"""

query_job = client.query(sql_query)
result_df = query_job.to_dataframe()
```

Output:

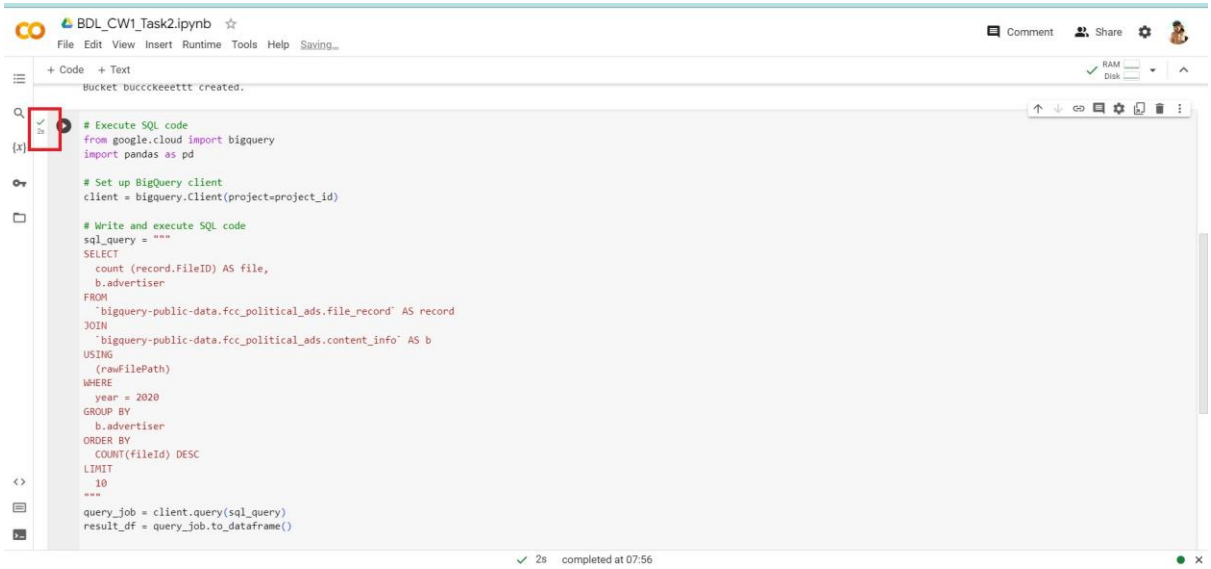

Figure 24: Running SQL query in google colab

Code:

After executing sql query its time to save that result into the bucket as a csv file. First the name of csv file was given and upload that file with the help of python blob function and print the result as successful meassage.

```
# Specify the file name for the CSV file in the bucket
file_name = "result.csv"

# Save the result to CSV and upload to Google Cloud Storage
result_df.to_csv(file_name, index=False)
blob = bucket.blob(file_name)
blob.upload_from_filename(file_name)

# Display result of files saved in google cloud bucket

print(f"Result saved to {cloud_storage_bucket_name}/{file_name}")
```

Output:

Figure 25: Saving csv files in google cloud

Let see the recently created '**result.csv**' files inside our google bucket '**buccckeeettt**'.

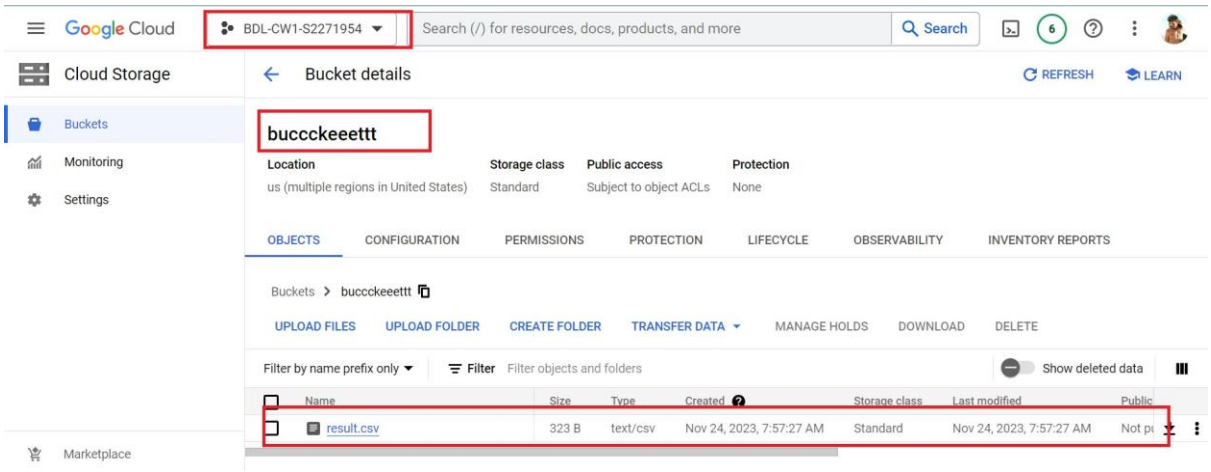

Figure 26: CSV files in google cloud

This is how we can use the google colab to run the sql queries and save the result into the google cloud storage bucket.

3 Task 3

There are various tools that are available in market to visualize the data. Though here we are going to use one of the most popular google visualize tools that Google Looker Studio.

First go the looker studio (https://lookerstudio.google.com/navigation/reporting) and click on blank report and select how you obtain your data. Here we are using our csv file that was save in google cloud storage bucket, so select Google Cloud Storage.

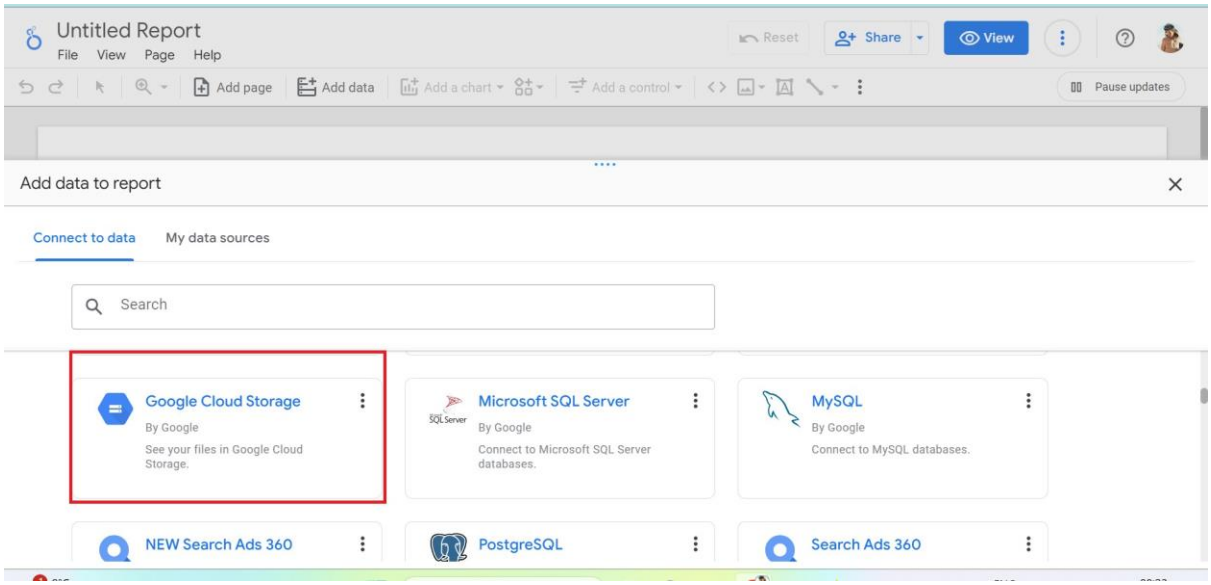

Figure 27: Selecting google cloud storage

After selecting you input data source we have to give file path of our file that is going to be used. Here the path was given and click on Add.

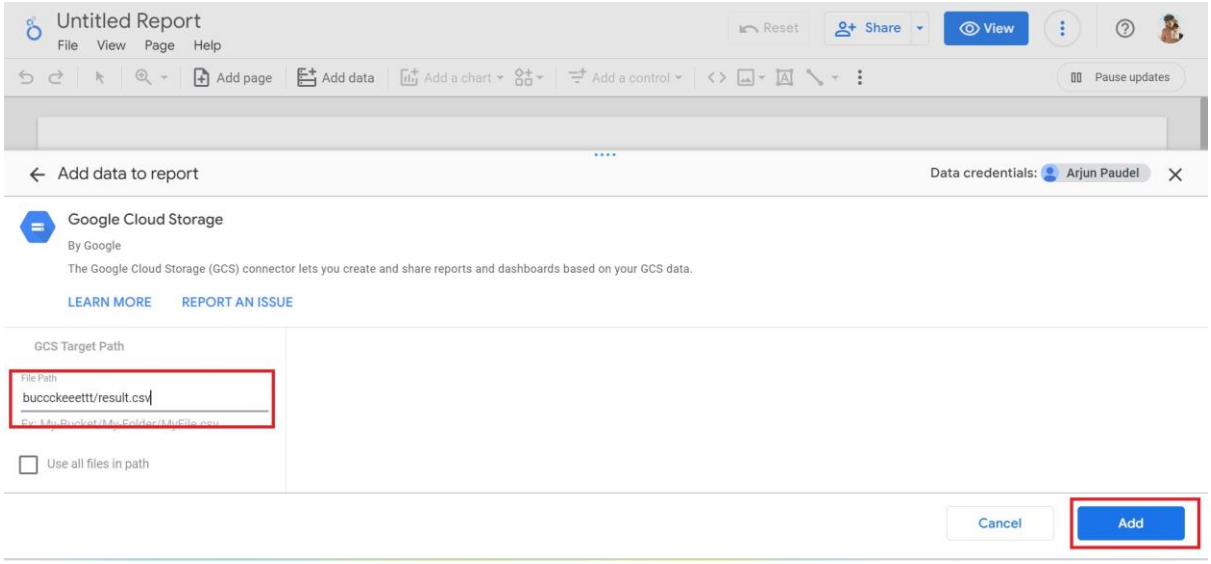

Figure 28: Giving direction of files

Once the file was loaded it will directly take us into studio where we can adjust different values according to requirements.

According to the data I had created three different type of graph and they are table graph, pie chart, and column chart.

Table Chart :-

Number of Files Uploaded by Advertiser

	advertiser	file
1.	Biden for President	38
2.	Elizabeth Warren for President-D (133004)	29
3.	Biden/D/President	25
4.	POL/Joe Biden/President/US/Dem	23
5.	Tom Steyer for President (1472)	21
6.	POL/Elizabeth Warren/President/US/Dem	21
7.	Warren For President	19
8.	Elizabeth Warren for President-D	17
9.	Biden For President	17
10.	POLI/J BIDEN/D/PRE/US	13

1 - 10 / 10

Figure 29: Visualization of data as table chart in looker studio

Column Chart :-

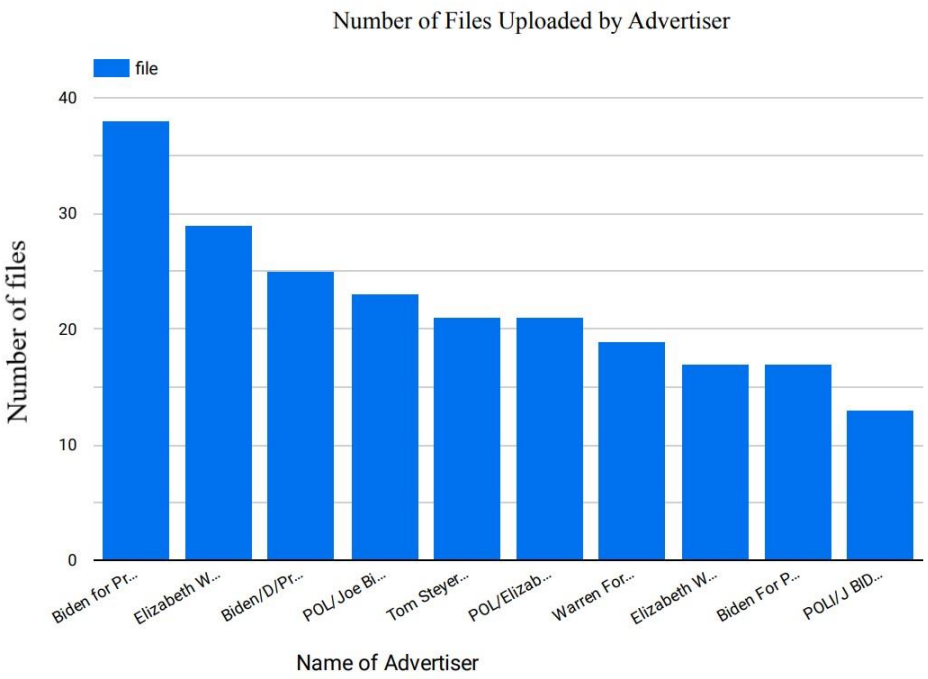

Figure 30:Visualization of data as column chart in looker studio

Pie Chart :-

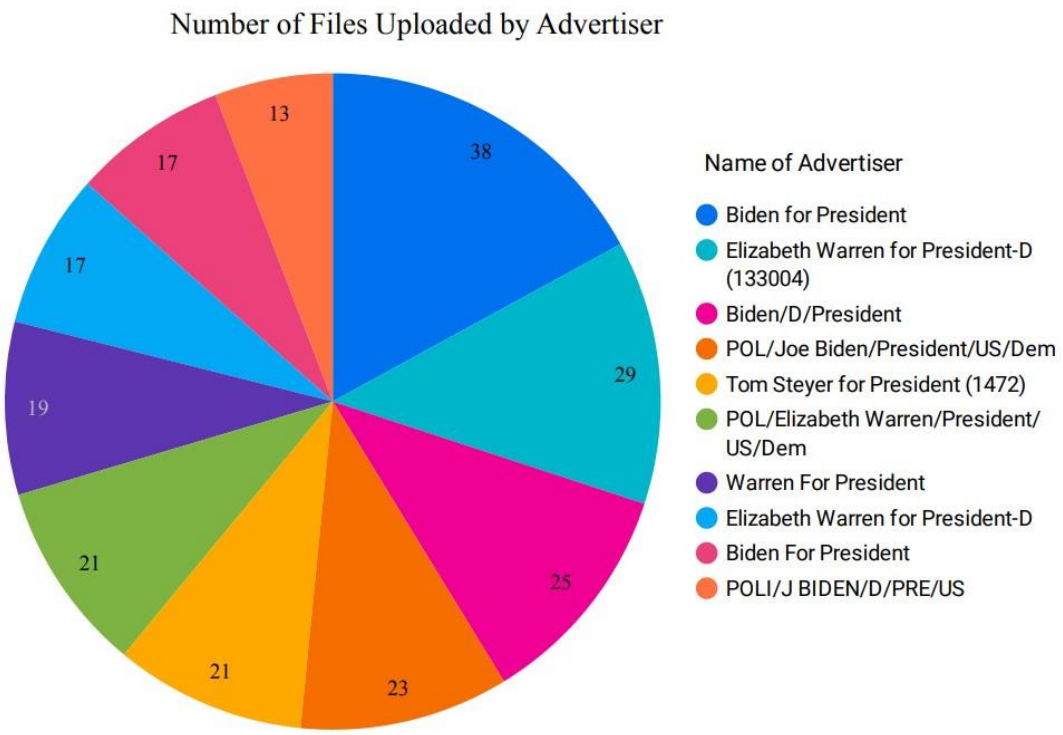

Figure 31:Visualization of data as pie chart in looker studio

This is how we can easily visualize our data through looker studio.